# Przetwarzanie danych walutowych

Tomasz Lama Łukasz Kita Łukasz Jezapkowicz

07.06.2020

# 1 Podstawowe pojęcia

Zanim przejdziemy do przedstawienia konkretnych algorytmów musimy wprowadzić kilka podstawowych terminów.

## 1.1 Arbitraż

Arbitraż w finansach i ekonomii to transakcja mająca na celu wykorzystanie różnicy w cenach tego samego dobra na dwóch różnych rynkach lub na tym samym rynku, lecz w dwóch różnych postaciach.
Dokonując arbitrażu inwestor dokonuje jednoczesnego zakupu i sprzedaży tego samego dobra po dwóch różnych cenach osiągając zysk wynikający z różnicy cen. Ze względu na jednoczesność zakupu i sprzedaży, wynik transakcji arbitrażowej jest z góry znany. Transakcje arbitrażowe są więc wolne od ryzyka.
Dana transakcja jest transakcją arbitrażową jedynie wtedy, gdy zakup i sprzedaż odbywają się równocześnie. Jeśli występuje pomiędzy nimi istotny odstęp czasowy, transakcja nie jest zaliczana do arbitrażowych.
W tej pracy przyglądamy się z bliska arbitrażowi wykonywanemu na walutach (arbitraż walutowy). By przybliżyć to pojęcie posłużymy się przykładem.
Załóżmy, że w Warszawie ceny walut wyglądają następująco: 1 $EUR = 2\ USD$ a w Krakowie 1 $EUR = 1.9\ USD$. Przeprowadzając arbitraż, inwestor mógłby kupować Euro za dolary w Krakowie i sprzedawać w Warszawie, za każdym razem uzyskując 10 $USD$ na każdych 100 $EUR$.

## 1.2 Konwergencja cen

Arbitraż powoduje zjawisko konwergencji cen. Jeśli na przykład na dwóch rynkach danego dobra wystąpi różnica w cenach, arbitrażyści będą kupować dobro na rynku, na którym jest ono tańsze powodując wzrost popytu i wywołując presję na wzrost cen, oraz sprzedawać na rynku gdzie jest ono droższe, zwiększając popyt i obniżając ceny. Powoduje to, że ceny na obu rynkach zaczynają się zbliżać do siebie tak długo, aż różnica staje się na tyle mała, że arbitraż przestaje być opłacalny.

## 1.3 Forex

Forex to tylko jedna z kilku nazw określających międzynarodowy rynek wymiany walut (używa się również określeń Foreign Exchange Market lub FX Market). Mówiąc najprościej - FOREX to rynek na którym wymienia się jedną walutę na drugą po ustalonej cenie. Jest to największy i najbardziej płynny rynek na świecie, a jego dzienne obroty przekraczają 5 bilionów dolarów. Głównymi handlującymi na rynku Forex są banki i inne duże instytucje finansowe, ale w zasadzie każdy z nas może być pośrednim uczestnikiem obrotu, np. kiedy dokonuje transakcji zakupu waluty kraju do którego wybiera się na wakacje. Pierwotnie główną funkcją rynku Forex było usprawnienie procesu wymiany walut na potrzeby międzynarodowego handlu surowcami, towarami czy usługami, ale jego bardzo duża zmienność i płynność sprawiła, że stał się niezmiernie atrakcyjnym miejscem inwestycji i spekulacji traderów z całego świata.
Jest to jedno z wielu miejsc, gdzie możemy wykorzystać zjawisko arbitrażu w praktyce. W rzeczywistości, znalezienie sytuacji, w której możemy wykorzystać arbitraż jest bardzo trudne (dlatego większość arbitrażystów to wieloletni, doświadczeni inwestorzy).

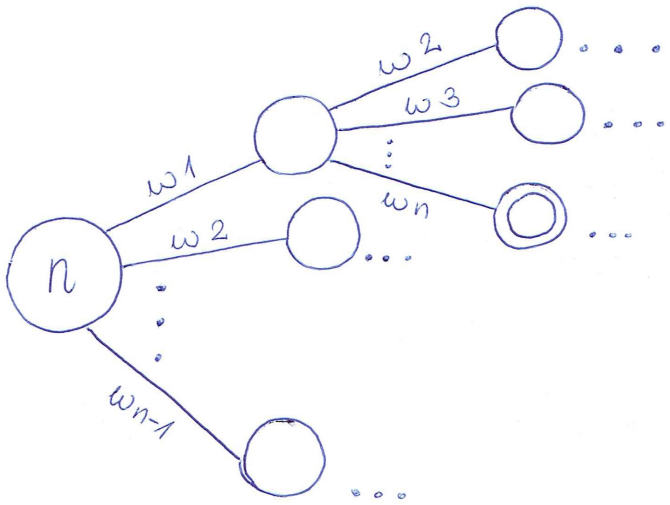
# 2 Algorytmy

## 2.1 Problem 1

Nasz problem sformułowany jest następująco. Dane jest $n$ walut oraz tablica o rozmiarze $n \times n$ symbolizująca kursy walutowe. Zaprojektuj algorytm, który pozwoli osiągnąć maksymalny zysk handlując walutami (wracając do wyjściowej waluty). Zakładamy, że każda wymiana może być dokonana tylko raz.

### 2.1.1 Algorytm naiwny

Najprostszym sposobem rozwiązania powyższego problemu jest metoda "brute force". Stworzymy drzewo o głębokości $n$ oraz czynniku rozgałęzienia $n-1$. Wierzchołki drzewa symbolizować będą waluty, zaś krawędzie kursy walutowe. Korzeniem drzewa będzie nasza startowa waluta $X$. Z każdego wierzchołka wychodzić będzie $n-1$ krawędzi do innych walut (waga krawędzi symbolizuje dany kurs walutowy). Wierzchołki symbolizujące walutę $X$ (różne od korzenia) oznaczamy specjalną flagą. Przykładowe drzewo widać poniżej.




Budowane drzewo

Po zbudowaniu drzewa należy znaleźć wszystkie ścieżki prowadzące z korzenia do specjalnie oznaczonych wierzchołków (wracamy do tej samej waluty). Dla każdej ścieżki obliczamy współczynnik $w = w_{i_1} * w_{i_2} * ... * w_{i_{k-1}} * w_{i_k}$ i wybieramy wszystkie ścieżki o wartości $w$ większej od 1. Każda taka ścieżka to osobna możliwość arbitrażu. Spośród wszystkich takich ścieżek (o ile istnieją) wybieramy tą o największej wartości $w$.

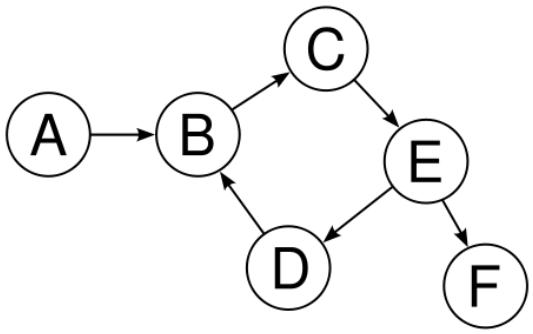
**Złożoność algorytmu:** Zarówno złożoność obliczeniowa jak i pamięciowa powyższego algorytmu to $O(n^n)$. Zauważmy, że możemy zredukować złożoność pamięciową do $O(n)$ używając prostej metodyki:
- jeśli dojdziemy do wierzchołka symbolizującego walutę $X$, nie generujemy jego dzieci.
- żeby zredukować współczynnik rozgałęzienia z $n-1$ do 1 możemy podczas generowania dzieci wybrać dziecko, które daje największy współczynnik $w$ do tej pory. W tym celu musimy w każdym wierzchołku przechowywać tymczasową wartość współczynnika $w$.

**Zastosowanie** Algorytm naiwny ma bardzo wąskie zastosowanie i nie jest stosowany w żadnym realnym programie. Wynika to z bardzo dużej złożoności czasowej budowy takiego drzewa. Na komputerze wykonującym $10^9$ bazowych operacji na sekundę budowa takiego drzewa dla $n = 20$ zajęłaby ponad 3 miliardy lat.

### 2.1.2 Szybszy algorytm

Powyższy algorytm nas nie zadowala (przez swoją złożoność). Możemy do naszego problemu podejść inaczej. Zbudujemy skierowany graf, którego wierzchołki symbolizować będą nasze waluty zaś krawędzie konkretne kursy walutowe ($w_i$). W tak zbudowanym grafie naszym zadaniem jest znaleźć wszystkie cykle, które spełniają zależność:

$$w_{i_1} * w_{i_2} * w_{i_3} * ... * w_{i_k} > 1$$




Przykład cyklu: $B \to C \to E \to D \to B$

Powyższy warunek jest stosunkowo trudny do rozwiązania, możemy go jednak przekształcić do bardziej interesującej formy:

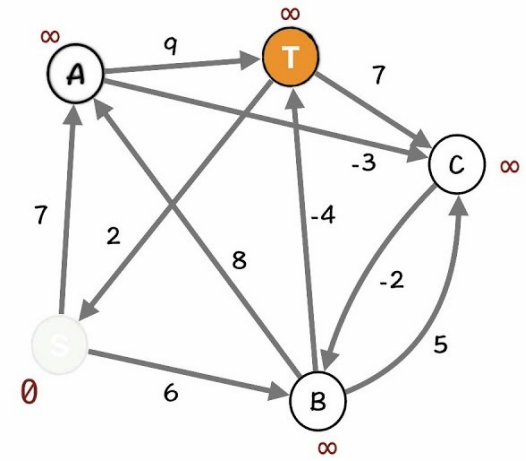
$$log(w_{i_1}) + log(w_{i_2}) + log(w_{i_3}) + ... + log(w_{i_k}) > 0$$

$$(-log(w_{i_1})) + (-log(w_{i_2})) + (-log(w_{i_3})) + ... + (-log(w_{i_k})) < 0$$

Takie przekształcenia dały nam dwie ważne rzeczy. Po pierwsze, zamiast produktu wag liczymy ich sumę, a po drugie nasze zadanie sprowadziło się do znalezienia cyklu o ujemnej sumie (odpowiednio przeskalowanych) wag krawędzi. Okazuje się, że w świecie algorytmów grafowych istnieje powszechnie używany algorytm Bellmana-Forda wykrywający takie cykle w zadowalającym (wielomianowym) czasie.

**Bellman Ford** By zrozumieć jak ma nam pomóc algorytm Bellmana-Forda należy najpierw poznać jego działanie. Wprowadźmy następujące oznaczenia:
$G(V, E)$ - graf z wierzchołkami $V$ oraz krawędziami $E$
$w(x)$ - waga wierzchołka $x$
$w(i, j)$ - waga krawędzi od wierzchołka $i$ do wierzchołka $j$
$p(j)$ - poprzednik wierzchołka $j$

Algorytm Bellmana-Forda przyjmuje wierzchołek wejściowy (source) i odnajduje najkrótsze ścieżki do wszystkich innych wierzchołków w $G$.
Gdy algorytm kończy swoje działanie, wszystkie wierzchołki przechowują szukaną wagę najkrótszej ścieżki jako $w(x)$. Działanie algorytmu opiera się na zasadzie relaksacji krawędzi. Krawędź $u \to v$ jest relaksowana jeżeli spełniony jest warunek $w(u) + w(u, v) < w(v)$. W takim wypadku zamieniamy wartość $w(v)$ na $w(u)+w(u, v)$. Po wykonaniu $k$ iteracji algorytm odnajduje wszystkie najkrótsze ścieżki składające się z maksymalnie $k$ krawędzi. Algorytm kończy swoje działanie po $|V - 1|$ iteracjach (najdłuższa ścieżka może składać się maksymalnie z $|V - 1|$ krawędzi).
Jeżeli po wykonaniu algorytmu nadal możemy zrelaksować pewną krawędź, to jest to równoznaczne z pojawieniem się ujemnego cyklu w grafie. Taki cykl możemy łatwo odtworzyć dzięki zapisywanym wartościom poprzedników wierzchołków w $p$.




```
Bellman-Ford ( G, w, s )
Initialize-Single-Source ( G, s )
for i = 1 to | G.V | - 1
    for each edge ( u, v ) ∈ G.E
        Relax ( u, v, w )
for each edge ( u, v ) ∈ G.E
    if v.d > u.d + w ( u, v )
        return FALSE
return TRUE
```

Żeby zastosować algorytm Bellmana-Forda dla naszego problemu wystarczy jedynie wykonać mały preprocessing wag. Wystarczy jedynie zmienić wartość każdej wagi zgodnie z formułą:

$$w_i = -log(w_i)$$

Przy takich wagach wystarczy użyć algorytmu Bellmana-Forda w celu znalezienia ujemnych cykli, a co za tym idzie, możliwości arbitrażu.

**Złożoność algorytmu:** Opisany algorytm posiada złożoność pamięciową $O(|V|)$ oraz złożoność czasową $O(|V| * |E|)$.

**Zastosowanie** Powyższy algorytm można stosować w wielu dziedzinach związanych z rynkiem wymiany walut (np. Forex'em). Osiągnięta złożoność pozwala skutecznie opracowywać strategię arbitrażu dla niemal dowolnej ilości wejściowych walut. Dla $n = 20$ nasz algorytm na wcześniej opisanym komputerze zająłby nieco ponad mikrosekundę. Dla $n = 150$ (istnieje mniej więcej tyle używanych walut) zająłby nieco ponad trzy milisekundy. W czasie wcześniej podanych 3 miliardów lat potrafiłby obliczyć rozwiązanie dla $n \approx 4,55 * 10^8$.

## 2.2 Problem 2

Opisywany w tej sekcji problem stanowi rozszerzenie problemu pierwszego. Zadany jest N-elementowy zbiór walut C oraz zbiór uporządkowanych krotek postaci ($c_i$, $c_j$, $p_{ij}$, $l_{ij}$) gdzie $c_i$, $c_j \in$ C, $p_{i,j} \in \mathbb{R}$ to kurs wymiany $c_j$ do $c_i$(ile jednostek waluty $c_j$ można kupić za jedną jednostkę waluty $c_i$), natomiast $l_{i,j} \in \mathbb{R}$ to maksymalna liczba jednostek waluty $c_i$, którą można wymienić na walutę $c_j$. Zasadność ograniczenia opisywanego przez $l_{i,j}$ wynika z faktu, że na giełdzie wymiany walut ilość pieniędzy w danej gotówce jest wartością skończoną. Nawet w wypadku, gdy ilość tych pieniędzy jest potencjalnie bardzo duża to należy zwrócić uwagę na fakt, że rynek jest w stanie zaoferować po danek cenie wymiany określoną ilość pieniędzy, a każda wymiana ma wpływ na stan rynku i może powodować zmianę ceny.
Podobnie jak w przypadku pierwszego opisywanego problemu będziemy starać się znaleźć okazję do arbitrażu - czyli znaleźć taki ciąg wymian, rozpoczęty w pewnej walucie, aby po jego wykonaniu otrzymać w startowej walucie większy kapitał niż na początku. Zakładamy, że podczas wymiany nie są pobierane żadne prowizje.

## 2.3 Rozwiązanie w oparciu o metodę programowania liniowego

Postaramy się tak zamodelować zadany problem, aby do jego rozwiązania możliwe było wykorzystanie metod znanych z rozwiązywania problemów programowania liniowego.

### 2.3.1 Definicja programowania liniowego.

Programowanie liniowe to problem optymalizacyjny, stanowiący specjalną klasę programowania matematycznego. Programowanie matematyczne to maksymalizacja funkcji celu f(x) przy dwóch rodzajach warunków ograniczających:

1. Pierwszego typu: h(x)=0
2. Drugiego typu: $g(x) \leqslant 0$

dla $x \in X \subset \mathbb{R}^n$ i funkcji f,h oraz g określonych na X. W przypadku programowania liniowego na funkcje f,g i h narzucone są warunki - otóż każda z tych funkcji jest funkcją liniową, to znaczy jest postaci:

$$F(x) = a_0 + a_1 x_1 + a_{2x2} + ... + a_n x_n$$

dla pewnych stałych $a_1, ..., a_n \in \mathbb{R}$

### 2.3.2 Sformułowanie problemu jako zadania programowania liniowego.

Ustalmy, że walutą początkową jest waluta oznaczona jako $c_0$. Niech $x_{i,j}$ oznacza ilość waluty $c_i$ wymienionej na $c_j$. Wprowadźmy następujące oznaczenia: niech $x_0$ oznacza kapitał początkowy (w walucie początkowej $c_0$), natomiast y kapitał końcowy (w walucie początkowej). Arbitraż można uznać za udany, jeżeli $y > x_0$. Zysk wypracowany w wyniku arbitrażu oznaczmy jako p. Z powyższego wynika zatem, że:

$$y = x_0 + p$$

W celu uproszczenia zapisu przyjmijmy, że indeks ostatniej waluty to n, co oznacza, że $n = N - 1$. Suma kapitału początkowego oraz całego kapitału uzyskanego na drodze wymian z innych walut

do waluty początkowej musi być równa sumie kapitału końcowego i ilości waluty początkowej $c_0$ wymienionej na wszystkie inne waluty, tj.:

$$x_0 + p_{1,0}x_{1,0} + ... + p_{n,0}x_{n,0} = y + x_{0,1} + ... + x_{0,n} \tag{1}$$

co po przekształceniu odpowiada:

$$x_0 = y + x_{0,1} + ... + x_{0,n} - (p_{1,0}x_{1,0} + ... + p_{n,0}x_{n,0}) \tag{2}$$

Celem będzie maksymalizacja wartości y, która odpowiada kapitałowi końcowemu w walucie początkowej.

Łatwo zauważyć, że w interesie podejmującego arbitraż jest, aby kapitał końcowy we wszystkich walutach innych niż waluta początkowa $c_0$ był równy zero. Wynika z tego zatem, że dla każdej waluty $c_k$, $k \neq 0$ zachodzi:

$$x_{k,0}+...+x_{k,k-1}+x_{k,k+1}+...+x_{k,n}-(p_{0,k}x_{0,k}+...+p_{k-1,k}x_{k-1,k}+p_{k+1,k}x_{k+1,k}+...+p_{n,k}x_{n,k}) = 0 \tag{3}$$

Powyższe równania będą stanowiły pierwszy typ ograniczeń nałożonych na rozwiązanie problemu. Z równania (2) oraz równań ograniczających (3) otrzymujemy następujący układ równań:

$$\begin{cases} y + x_{0,1} + ... + x_{0,n} - (p_{1,0}x_{1,0} + ... + p_{n,0}x_{n,0}) = x_0 \\ x_{1,0} + x_{1,2} + ... + x_{1,n} - (p_{0,1}x_{0,1} + p_{2,1}x_{2,1} + ... + p_{n,1}x_{n,1}) = 0 \\ x_{2,0} + x_{2,1} + x_{2,3} + ... + x_{2,n} - (p_{0,2}x_{0,2} + p_{1,2}x_{1,2} + p_{3,2}x_{3,2} + ... + p_{n,2}x_{n,2}) = 0 \\ \vdots \\ x_{n,0} + ... + x_{n,n-1} - (p_{0,n}x_{0,n} + ... + p_{n-1,n}x_{n-1,n}) = 0 \end{cases}$$

Ponadto, na rozwiązanie nałożono również drugi typ ograniczeń, w postaci nierówności:

$$\forall_{i\in\{0,...,n\}}\forall_{j\in\{0,...,n\}} \quad 0 \leqslant x_{i,j} \leqslant l_{i,j} \tag{4}$$

Zapisujemy problem w postaci macierzowej. Z uwagi na fakt, że dla ogólnego przypadku z n-walutami zapis ten byłby nieczytelny, zdecydowano się zapisać go dla n=2. Otrzymano:

$$\begin{bmatrix} 1 & 1 & 1 & -p_{1,0} & 0 & -p2,0 & 0 \\ 0 & -p_{0,1} & 0 & 1 & 1 & 0 & -p_{2,1} \\ 0 & 0 & -p_{0,2} & 0 & -p_{1,2} & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} y \\ x_{0,1} \\ x_{0,2} \\ x_{1,0} \\ x_{1,2} \\ x_{2,0} \\ x_{2,1} \end{bmatrix} = \begin{bmatrix} x_0 \\ 0 \\ 0 \end{bmatrix}$$

z warunkami:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leqslant \begin{bmatrix} x_{0,1} \\ x_{0,2} \\ x_{1,0} \\ x_{1,2} \\ x_{2,0} \\ x_{2,1} \end{bmatrix} \leqslant \begin{bmatrix} l_{0,1} \\ l_{0,2} \\ l_{1,0} \\ l_{1,2} \\ l_{2,0} \\ l_{2,1} \end{bmatrix}$$

Postaramy się teraz zamienić ograniczenia typu drugiego tak, aby dało się je przedstawić w postaci równości. W tym celu, mając daną nierówność:

$$a_1x_1 + a_2x_2 + ... + a_nx_n \leqslant b$$

przekształcamy ją do postaci:

$$x' = b - (a_1x_1 + a_2x_2 + ... + a_nx_n)$$

gdzie nowa zmienna $x'$ reprezentuje różnicę pomiędzy górnym ograniczeniem b, a rzeczywistą wartością funkcji ograniczającej $a_1x_1 + a_2x_2 + ... + a_nx_n$. Ostatecznie, daną funkcję ograniczającą możemy zapisać jako:

$$a_1x_1 + a_2x_2 + ... + a_nx_n + x' = b$$

Dodając nowe zmienne na potrzeby zapisania różnicy pomiędzy górnym ograniczeniem b, a wartością funkcji ograniczającej, od których nie zależy maksymalizowana funkcja, możemy zapisać w postaci równania macierzowego cały interesujący nas problem arbitrażu, z nałożonymi ograniczeniami pierwszego i drugiego typu, jako:

$$\begin{bmatrix}
1 & 1 & 1 & -p_{1,0} & 0 & -p_{2,0} & 0 & 0&0&0&0&0&0 \\
0 & -p_{0,1} & 0 & 1 & 1 & 0 & -p_{2,1} & 0&0&0&0&0&0 \\
0 & 0 & -p_{0,2} & 0 & -p_{1,2} & 1 & 1 & 0&0&0&0&0&0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1&0&0&0&0&0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0&1&0&0&0&0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0&0&1&0&0&0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0&0&0&1&0&0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0&0&0&0&1&0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0&0&0&0&0&1
\end{bmatrix} \cdot \begin{bmatrix}
y \\ x_{0,1} \\ x_{0,2} \\ x_{1,0} \\ x_{1,2} \\ x_{2,0} \\ x_{2,1} \\ x'_{0,1} \\ x'_{0,2} \\ x'_{1,0} \\ x'_{1,2} \\ x'_{2,0} \\ x'_{2,1}
\end{bmatrix} = \begin{bmatrix}
x_0 \\ 0 \\ 0 \\ l_{0,1} \\ l_{0,2} \\ l_{1,0} \\ l_{1,2} \\ l_{2,0} \\ l_{2,1}
\end{bmatrix}$$

### 2.3.3 Rozwiązanie problemu

Rozwiązania tak zapisanego problemu można przy użyciu standardowych metod używanych w programowaniu liniowym, np. poprzez algorytm sympleksowy. Po wykonaniu algorytmu sympleksowego otrzymamy rozwiązanie w postaci wektora:

$$\left[\ y,\ x_{0,1},\ x_{0,2},\ x_{1,0},\ x_{1,2},\ x_{2,0},\ x_{2,1},\ x'_{0,1},\ x'_{0,2},\ x'_{1,0},\ x'_{1,2},\ x'_{2,0},\ x'_{2,1}\ \right]^T$$

z którego będzie nas interesowała tylko część:

$$\left[\ y,\ x_{0,1},\ x_{0,2},\ x_{1,0},\ x_{1,2},\ x_{2,0},\ x_{2,1}\ \right]^T$$

gdzie y będzie kapitałem w walucie początkowej $c_0$ po wykonaniu wymian. Element $x_{i,j}$ wektora odpowiada ilości pieniędzy w walucie $c_i$, które należy wymienić na $c_j$. Zakładamy, że wymiany zachodzą jednocześnie (tzn. tak, jakbyśmy mieli zgromadzoną odpowiednią liczbę środków w danych walutach), jednak po wykonaniu wszystkich wymian suma kapitału zgromadzonego w innej walucie niż początkowa jest równa zero (Nie zagłębiamy się tutaj w samą kolejność wykonywania tych operacji).

### 2.3.4 Wnioski

Złożoność obliczeniowa algorytmu sympleksów jest duża i w pesymistycznym przypadku liczba obliczeń wynosi: $2^d$, gdzie d to liczba zmiennych w szukanym wektorze. W naszym przypadku:

$$d = 1 + 2 \cdot (N^2 - N)$$

gdzie N to liczba walut. W rzeczywistości algorytm sympleksów daje rezultaty w zdecydowanie krótszym czasie, ponadto w każdym kroku można zweryfikować, czy arbitraż rzeczywiście zachodzi i zdecydować się na ewentualne jego wykonanie.

Kluczowym problemem było takie przedstawienie problemu arbitrażu, aby możliwe było jego zapisanie w postaci problemu programowania liniowego. Dodatkowe skomplikowanie modelu ograniczeniami związanymi z maksymalną ilością pieniędzy, które po danej cenie można wymienić sprawia, że taki model może być stosowany w rzeczywistych systemach szukających okazji do arbitrażu. Ponadto w dość prosty sposób można do tego modelu nałożyć dodatkowe ograniczenia (takie jak prowizja giełdy za każdą transakcję).

# 3 HFT - handel wysokich częstotliwości

Oprócz znalezienia odpowiednich walut do zakupu i sprzedaży, równie ważny, jeśli nie ważniejszy jest czas wykonania tej operacji. Sekunda zwłoki może skutkować, że dana transakcja zostanie przeprowadzona przez konkurencję. Czas trwania procesu potrzebnego do zawarcia transakcji skrócił się z kilku-kilkunastu minut do kilku sekund. Jednak, gdy chcemy zarobić na minimalnych wahaniach cen walut/akcji, musimy te operacje przeprowadzać w przeciągu ułamków sekund.

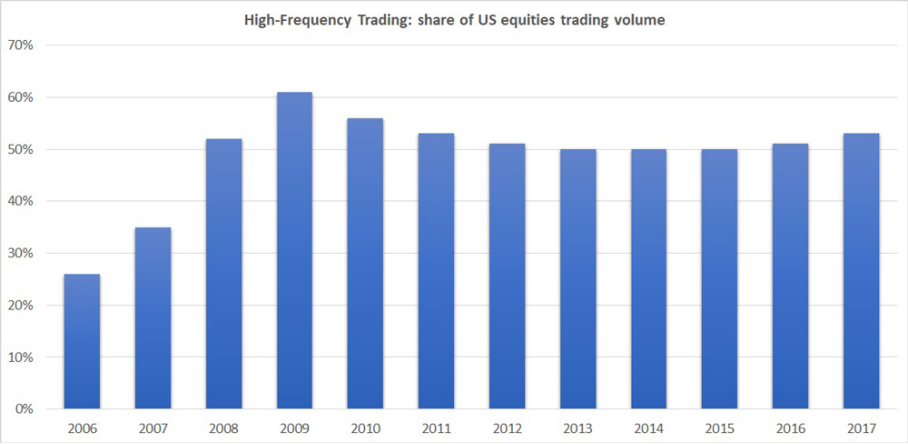
## 3.1 Początki

Na przełomie 2004 i 2005 roku na światowych giełdach zaczęły pojawiać się pierwsze transakcje zawarte poprzez wykorzystanie tzw. High Frequency Trading (HFT). HFT, czyli handel wysokich częstotliwości, jest strategią inwestycyjną polegającą na zawieraniu wielu transakcji giełdowych w ciągu ułamków sekund. Dzięki wykorzystaniu ultraszybkiego transferu danych oraz skomplikowanych algorytmów decyzyjnych, komputery wykorzystujące HFT mogą zawierać setki transakcji w ciągu milisekund, wyprzedzając konkurencję. Zysk na każdej z tych transakcji najczęściej wynosi ułamkowe części groszy bądź jeden pips (w przypadku Forex). Jednak biorąc pod uwagę ilość transakcji, które mogą być zawarte w ciągu sekundy, kwota zyskana na transakcjach jest znacznie większa niż można by przypuszczać. Dobry trader jest w stanie wykonać kilka – kilkanaście transakcji w ciągu minuty. Automat HFT – kilka tysięcy w ciągu sekundy. W tej dziedzinie szanse człowieka w starciu z maszyną wypadają mizernie.

Bycie szybszym o ułamek sekundy niż inni ma również inne zalety i pozwala uzyskiwać zyski na transakcjach arbitrażowych. Arbitraż wykorzystuje nieznaczne różnice w cenach tego samego aktywa notowanego na różnych giełdach. Kupno akcji po niższej cenie w kraju A i ich natychmiastowa sprzedaż po wyższej cenie w kraju B stanowi okazję na zysk bez ryzyka. Takie sytuacje są błyskawicznie zauważane przez pozostałych uczestników rynku i momentalnie niwelowane. HFT pozwala na ich wykorzystanie wyprzedzając tradycyjnych traderów.

## 3.2 Rozwój technologi

Gdy HFT zaczął odnosić (i powtarzać) swoje pierwsze sukcesy, wieść o "automatach inwestycyjnych" lotem błyskawicy obiegła świat. Handel wysokich częstotliwości szybko zyskał popularność w USA, czego dowodem jest podwojenie udziału HFT w obrocie akcjami w ciągu 2 lat (z 25% do ponad 50% w latach 2006 -2008). Szczyt zainteresowania przypadł na rok 2009, gdy udział ten wyniósł ponad 60%. Kryzys finansowy nadszarpnął nieco zaufanie do rynków jednak w ostatnich latach udział handlu wysokich częstotliwości w USA utrzymuje się na stabilnym poziomie około 50%.




Rysunek 1: Procentowy udział HFT w transakcjach w USA

Inwestorzy opierając się na robotach wykorzystują najlepszą technologię, jeżeli wychodzi nowy procesor, to kupują go jako pierwsi. W takim handlu nikt nie oszczędza na programistach oraz technologii. Małe opóźnienie może przesądzić, że ktoś inny nas wyprzedzi i zgarnie cały zysk z rynku. Dzięki temu w Stanach Zjednoczonych liczy się nie tylko technologia, ale położenie infrastruktury. Miejsca obok serwera giełdy są naprawdę pożądane. Serwer giełdy New York Stock Exchange położony jest 50 km od Wall Street. W tym samym budynku lub bliskiej okolicy (najlepiej najbliżej jak się da) firmy HFT mają swoje siedziby. Zasada jest prosta, ten, kto znajduje się bliżej giełdy będzie szybszy od innych i to wystarczy. Każdy centymetr bliżej serwera, skutkuje szybszą wymianą danych, co przekłada się na wykonanie transakcji przed konkurencją.

# 4 Podsumowanie

Wykorzystując skomplikowane algorytmy i najnowocześniejsze technologie, jesteśmy wstanie zarobić na arbitrażu. W tym biznesie liczy się każda milisekunda. Jest to spowodowane faktem, że konkurencja jest duża. Firmy prześcigają się w rozwiązaniach, nie oszczędzając na sprzęcie oraz naukowcach opracowujących stosowane algorytmy, jak również programistach je implementujących. Pierwszy przedstawiony algorytm jest niezwykle szybki. Jego wadą jest fakt, że nie uwzględnia tego, że nasz zasób gotówki jest ograniczony, co ogranicza sens jego stosowania do sytuacji, gdzie operujemy na niewielkich kwotach, których wymiana nie wpływa na sytuację na rynku. Drugi algorytm uwzględnia ograniczenia związane ze skończonością zasobów i po pełnym wykonaniu zawsze przedstawia najkorzystniejsze rozwiązanie, jednak czas jego działania jest długi, z uwagi na co przy jego wykorzystaniu trzeba czasem rozważać podejście heurystyczne, skutkujące w podaniu rozwiązania poprawnego, lecz niekoniecznie najbardziej korzystnego. W części dotyczącej HFT zostało

przedstawione w jaki sposób, wykorzystując wcześniej opisane algorytmy, jednak ze znaczną liczbą modyfikacji i większym stopniem skomplikowania, w rzeczywistości przeprowadzić taki arbitraż. W obecnym świecie, z uwagi na fakt niemal pełnej informatyzacji systemów wymiany walut, ceny na różnych giełdach zbiegają do takiej samej kwoty w niezwykle krótkim okresie. Jest to bezpośredni skutek arbitrażu dokonywanego na tych giełdach. W celu uzyskania zarobku związanego z arbitrażem trzeba zatem działać bardzo szybko, aby zmieścić się w przedziale czasu, w którym występują różnice w cenach.

# 5 Bibliografia

https://www.investopedia.com/ask/answers/what-is-arbitrage/
https://forexrev.pl/czym-jest-arbitraz/
https://stackoverflow.com/questions/2282427/interesting-problem-currency-arbitrage
https://medium.com/@anilpai/currency-arbitrage-using-bellman-ford-algorithm-8938dcea56ea
https://towardsdatascience.com/algorithm-shortest-paths-1d8fa3f50769
https://www.ig.com/en-ch/trading-strategies/high-frequency-trading-explained–why-has-it-decreased–181010
https://admiralmarkets.pl/education/articles/automated-trading/roboty-hft "A Discussion of Linear Programming and its Application to Currency Arbitrage Detection", Rachel Smith, University of Redlands, 2013,
dostęp 9.06.2020: https://inspire.redlands.edu/cgi/viewcontent.cgi?article=1754context=cas_honors